King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering
Information and Computer Science Department
First Semester 181 (2018/2019)

ICS 202 – Data Structures
Major Exam 1
Sunday, September 30th, 2018
Time: 90 minutes

Name: _____**Key**_____

ID# | | | | | | | | | |

Section 01 Dr. Sami
Section 02 Dr. Irfan

Question #	Max Marks	Marks Obtained
1	25	
2	25	
3	25	
4	25	
Total	100	

Instructions

1. Write your name and ID in the respective boxes above and circle your section.
2. This exam consists of 5 pages, including this page, plus one double-sided reference sheet.
3. This exam consists of 4 questions. You have to answer all the 4 questions.
4. The exam is closed book and closed notes. No calculators or any helping aids are allowed.
5. Make sure you turn off your mobile phone and keep it in your pocket if you have one.
6. The questions are not equally weighed.
7. The maximum number of points for this exam is 100.
8. You have exactly 90 minutes to finish the exam.
9. Make sure your answers are readable.
10. If there is no space on the front of the page, feel free to use the back of the page. Make sure you indicate this in order not to miss grading it.

Q.1 (20 points) For the following piece of code:

```
public static int complex(float [] array, int n , int m) {
  for (int i=1; i<=n*n; i*=4)
    for (int j=1; j<i; j++)
        array[j] = array[j]+1; // MyStatement1

  for (int i=4; i<=n-4; i+=4)
    for (int j=i-4; j<i+4; j++)
        array[j] = array[j]+1; // MyStatement2

}
```

a) (10 points) Count the number of times MyStatement1 gets executed in terms of n.

a) (10 points) Count the number of times MyStatement1 gets executed in terms of n.

outer loop: # values of i: $1 \quad 4 \quad 16 \quad \dots \quad n^2$

$4^0 \quad 4^1 \quad 4^2 \quad \dots \quad 4^r$ $\qquad r = \log_4 n^2$

$$T(n) = \sum_{s=0}^{\log_4 n^2} \sum_{j=1}^{4^s - 1} 1 = \sum_{s=0}^{\log_4 n^2} 4^s - 1$$

$$= \sum_{s=0}^{\log_4 n^2} 4^s - \sum_{s=0}^{\log_4 n^2} 1 = \frac{4^{\log_4 n^2 + 1} - 1}{3} - \log_4 n^2$$

$$= \frac{1}{3}\left(4n^2 - 1\right) - \log_4 n^2$$

$$= \boxed{\frac{4}{3}n^2 - \frac{1}{3} - 2\log_4 n}$$

b) (10 points) Count the number of times MyStatement2 gets executed in terms of n.

outer loop: # values of i: $4 \quad 8 \quad 12 \quad 16 \quad \dots \quad n-4$

$4\times1 \quad 4\times2 \quad 4\times3 \quad 4\times4 \quad \dots \quad 4\times r$ $\qquad r = \frac{n-4}{4}$

$$T(n) = \sum_{s=1}^{\frac{n-4}{4}} \sum_{j=4s-4}^{4s+3} 1$$

$$= \sum_{s=1}^{\frac{n-4}{4}} \left(4s+3-4s+4+1\right) = \sum_{s=1}^{n-4/4} 8 = 8\left(\frac{n-4}{4} - 1 + 1\right)$$

$$= \boxed{2n - 8}$$

c) (5 points) Determine Big O complexity of Complex(n)

$O(n^2)$

Q.2 (25 points): Complexity Analysis

a) (7 points) If the number of basic operations in an algorithm is given by:

$$2n^2 + 100\ n \log n + 300n + 2000$$

Write the Big-O complexity of the algorithm. Show your steps/justification.

The max term is $O(n^2)$ as $O(n^2) > O(n \log n) > O(n)$
Thus, Big-O complexity of the algorithm is $O(n^2)$

b) (8 points) If the number of basic operations in algorithm–1 is given by ***n+10*** and for algorithm–2 is ***n/2+100***, both the algorithms are O(n). Does this mean that both the algorithms will take the same amount of time in solving a given problem and it does not matter which algorithm we choose? Explain your answer.

No, both the algorithms will not take the same amount of time in solving a given problem.
For small n, (***n+10***) will run faster that (***n/2+100***). Precisely,
n+10 < n/2 + 100
n/2 < 90
n < 180. Thus for n <180, (***n+10***) will run faster that (***n/2+100***)

and for n > 180, (***n+10***) will run slower that (***n/2+100***)

c) (10 points) State the big-O complexity of each of these data structures for each of the following methods:

	addFirst (addToHead)	**addLast (addToTail)**	**deleteFirst (deleteHead)**	**deleteLast (deleteTail)**	**insert[1]**	**delete[2]**
SinglyLinkedList	**O(1)**	**O(1)**	**O(1)**	**O(n)**	**O(n)**	**O(n)**
DoublyLinkedList	**O(1)**	**O(1)**	**O(1)**	**O(1)**	**O(n)**	**O(n)**
Array implementation of List[3]	**O(n)**	**O(1)**	**O(n)**	**O(1)**	**O(n)**	**O(n)**

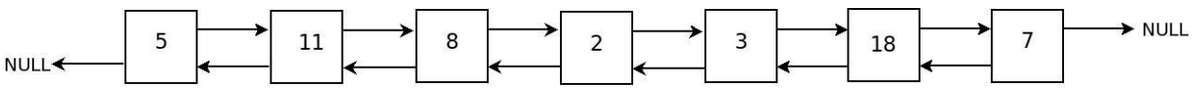
1. The insert method inserts a new element in a specific position (given as a parameter).
2. The delete method deletes a given element (given as parameter) from the list.
3. The Array implementation should not contain gaps (empty cells)

Q.3: (25 points) Consider the DLL and DLLNode classes:

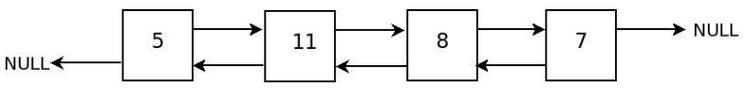
```
public class DLLNode<T> {
    public T info;
    public DLLNode<T> next, prev;
    …..
}

public class DLL<T> {
    private DLLNode<T> head, tail;
    …….
}
```

Knowing that you can use any method of the class DLL<T> (see the formulas sheet at the end of the exam), answer the following questions:

Provide a method **`deleteChunk`** which takes as parameter 2 integer indices i and j and deletes all nodes between the i'th and the j'th nodes.
Note: all special cases need to be treated by throwing exceptions.
Example: calling deleteChunk(4,7) on the following DLL



Results in the following DLL:




```
public void deleteChunk(int i, int j)
    {

      if (i < 0 || i > j || j > length())

           throw New Exception("Invalid input");

      if (isEmpty() || i == j)
           return;

        DLLNode<T> temp;

        for (int k = 0, temp = head;k<i; k++, temp = temp.next);

        if (m == length())
        {
          temp.next = null;
          tail = temp;
        }

        DLLNode<T> temp2;

        for (int k = i, temp2 = temp.next;k<j; k++, temp2 = temp2.next);

        temp.next = temp2;
        temp2.previous = temp;


    }
```

Q.4: (25 points) Provide an implementation of a method *mergeSorted()*, which takes two sorted lists as parameters and builds a sorted listed which combines both the list.
Note: All the lists are singly-linked lists.

```
public class SLLNode<T> {
     public T info;
     public SLLNode<T> next;
     .....
}

public class SLL<T> {
     private SLLNode<T> head, tail;
     ……..

public void mergeSorted(SLL<T> list1, SLL<T> list2) {
             SLLNode<T> tmp1 = list1.head;
             SLLNode<T> tmp2 = list2.head;

             while(tmp1!=null && tmp2!=null) {
                    if (((Integer)tmp1.info).compareTo((Integer)tmp2.info) <0) {
                           this.add2Tail(tmp1.info);
                           tmp1 = tmp1.next;
                    }
                    else {
                           this.add2Tail(tmp2.info);
                           tmp2 = tmp2.next;
                    }
             }// end while

             if(tmp1 == null) {
                    while(tmp2!=null){
                           this.add2Tail(tmp2.info);
                           tmp2 = tmp2.next;
                    }
             }

             if(tmp2 == null) {
                    while(tmp1!=null){
                           this.add2Tail(tmp1.info);
                           tmp1 = tmp1.next;
                    }
             }
   }// end method mergeSorted
```

Quick Reference Sheet

```
public class SLLNode<T> {
    public T info;
    public SLLNode<T> next;
  public SLLNode();
  public SLLNode(T el)
  public SLLNode(T el, SLLNode<T> ptr);
}

public class SLL<T> {
    protected SLLNode<T> head, tail;
  public SLL();
  public boolean isEmpty();
  public void addToHead(T el);
  public void addToTail(T el);
  public T deleteFromHead();
  public T deleteFromTail();
  public void delete(T el);
  public void printAll();
  public boolean isInList(T el);
}

public class DLLNode<T> {
    public T info;
    public DLLNode<T> next, prev;
  public DLLNode();
  public DLLNode(T el);
  public DLLNode(T el, DLLNode<T> n,
                       DLLNode<T> p);
}

public class DLL<T> {
    private DLLNode<T> head, tail;
  public DLL();
  public boolean isEmpty();
  public void setToNull();
  public void addToHead(T el);
  public void addToTail(T el);
  public T deleteFromHead();
  public T deleteFromTail();
  public void delete(T el);
  public void printAll();
  public boolean isInList(T el);
}
```

$$\sum_{i=m}^{n} c = c\left(\sum_{i=m}^{n} 1\right) = c.(n-m+1)$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}, a \neq 1$$

$$\log_b a = \frac{\ln a}{\ln b} \quad , \quad \log ab = \log a + \log b$$

$$\log \frac{a}{b} = \log a - \log b \quad , \quad a^{\log_a b} = b$$

$$\left(a^b\right)^c = \left(a^c\right)^b = a^{bc}$$